Binary Tumor Classifier: case study with standard machine learning algorithms

Riccardo Berra
University of Verona,
Department of Computer Science,
Master degree on Computer Engineering for Robotics and Smart industry,
riccardo.berra@studenti.univr.it,
Machine learning final project,
Academic year 2022-2023

Abstract - This paper focuses on the classification of brain tumors thanks to the application of advanced machine learning technologies. Usually, an accurate diagnosis and classification of brain tumors is fundamental to radiologists and oncologists in order to ensure a prompt diagnosis that might be life saving. In this context, Magnetic Resonance Imaging ("MRI") is universally recognized as the best available technique in order to detect brain tumors. However, the resulting MRI scan will require a manual reading, which might be subject to human error. Therefore, in this paper, we have conducted a study based on a public dataset containing 3,762 frontal brain MRI images, to which we have applied multiple machine learning methods (including PCA, KNN, SVM and CNN) in order to develop a classifier that is able to predict either (i) the presence or (ii) the absence of a brain tumor. Based on results, it emerges that "Convolutional Neural Networks" is the most accurate method (with 95% accuracy), compared for example to the KNN method (K=3) which has a 1% lower accuracy. The objective of this paper is to prove that the application of machine learning models (and of the CNN in particular) to MRI images reading, can be a valuable tool supporting healthcare professionals in the context of diagnosing brain tumors, reducing the risk of human error and allowing for more accurate and timely diagnoses.

Keywords: Brain tumor, Classification, Machine Learning

1 Introduction

Tumors, abnormal growths of cells, represent a formidable challenge to global healthcare systems [1], affecting millions of lives annually. The early and accurate diagnosis of tumors is pivotal for effective treatment, improving patient prognosis, and reducing the burden on healthcare resources. In this era of rapid technological advancements, machine learning algorithms have emerged as essential tools in the quest for more precise and efficient tumor classification. MRI is undoubtedly at the cornerstone of brain tumor imaging, playing a key role in all phases of patient journey, starting from the diagnosis, through therapy planning, to treatment response and / or recurrence assessment. [1] As accuracy is an important criterion for classification, computer vision researchers have developed several approaches, mainly based on deep learning, such as ResNet (2+1)D [2][3], ResNet mixed convolution [2][3], and ResNet3D[2][3]. These deep-learning based models have a remarkable capacity to discriminate and have progressively become an integral part of medical research and of healthcare. In this work, we will focus on simpler techniques that form the basis of classification fundamentals through machine learning and we will analyze how these behave as they are applied to a dataset of brain MRI scans.

2 Objectives

The aim of this paper is to analyze how a series of classic machine learning algorithms performs in the binary classification of brain tumors in Magnetic Resonance Imaging (MRI) images. Specifically, we have focused on a dataset of frontal brain MRI scans. This dataset contains images of both (i) healthy brains and (ii) brains affected by tumors of various sizes.

The aim is to create a classifier that is capable of predicting whether a given MRI image contains a tumor or not. In order to build the classifier, a series of standard machine learning algorithms have been considered: PCA ("Principal Component Analysis"), KNN ("K-Nearest Neighbors"), SVM ("Support Vector Machine"), and CNN ("Convolutional Neural Networks"). In this project, we will use PCA in order to reduce the dimensionality of the features and enable faster execution of the KNN and SVM algorithms. By using the KNN algorithm, the goal is to find the most effective value of K to allocate the image into one of the two classes. By testing three kernels (Linear, Polynomial, RBF), we will evaluate which one allows the SVM (Support Vector Machine) to optimally discriminate between the two classes. The final objective is to then use the dataset to train a simple CNN (Convolutional Neural Network) and to assess how well the model can correctly classify between the two classes.

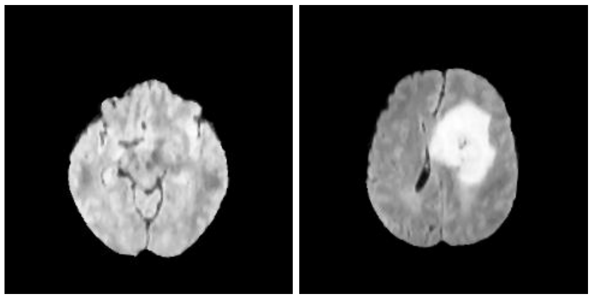
Fig. 1 Clean brain (left) and brain with tumor (right)

3 Methodology

In the development of machine learning algorithms, the choice of the dataset is one of the most critical and fundamental aspects for optimal model learning. In this project, we selected a publicly available dataset downloaded from the Kaggle website.

Version 1.18, September 14, 2023

3.1 Dataset. This dataset is composed of 3,762 brain scans obtained through MRI (**Magnetic Resonance Imaging**). The brain scans were performed frontally on both healthy subjects and subjects with brain tumors of various sizes. Specifically, these are black and white scans with a resolution of 240x240 pixels. Additionally, the dataset includes a metadata file containing 5 first-order features (**Mean, Variance, Standard Deviation, Skewness, Kurtosis**), 8 second-order features (**Contrast, Energy, ASM (Angular Second Moment), Entropy, Homogeneity, Dissimilarity, Correlation, Coarseness**), along with the column of image labels where **Tumor = 1** indicates the presence of a tumor, and **No tumor = 0** denotes a clean scan.

Given the volume of data and the features, it will be necessary for some algorithms to have a reduced feature space to enable fast computation during classification, as it is the case with SVM. In this scenario, **PCA (Principal Component Analysis)** is used in order to simplify a complex dataset.

3.2 PCA Principal Component Analysis. PCA is a technique used to reduce the dimensionality of complex datasets, and to increase interpretability while minimizing information loss [4]. This technique identifies the principal components, which are vectors that explain the maximum variance in the data. These vectors are projected onto a new coordinate basis, reducing the number of features of the dataset to a smaller number of principal components, aiming to preserve the maximum possible variance. In this work, PCA is applied to the images, technically a list of flat black and white image vectors of size $57600x1xTotalImages$. The resulting vector is then used as inputan for various classifiers (SVM and KNN).

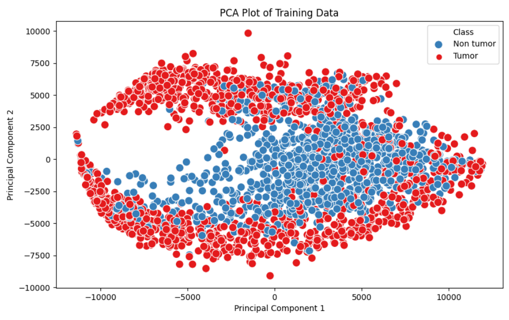

Fig. 2 PCA with 100 components on training set

3.3 SVM Support Vector Machine.. Feature reduction is crucial for classifiers like SVM due to high feature dimensionality [5]. SVM, which was introduced by Vapnik [6], aims to find an optimal hyperplane to separate classes, maximizing the margin between them. Kernel choice, mathematical functions mapping data into higher-dimensional spaces, is critical for SVM classifiers. In this paper, we utilize the following kernels among various types [7]:

- Linear.
- Polynomial.
- RBF (Radial Basis Function).

3.3.1 ***Linear Kernel..*** It is one of the simplest types of kernels used in SVM classifiers. It is used when it is assumed that the data is linearly separable in the original input space, allowing the creation of a decision hyperplane (a line in 2D, a plane in 3D, or a hyperplane in higher-dimensional spaces) that clearly separates different data classes.

It is represented as the inner product between two input vectors: $K(x, y) = x \cdot y$. Linear kernel SVMs are often used when it is believed that the data is already linearly separable or when a simpler and computationally efficient solution is desired.

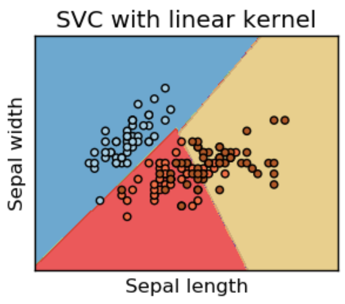

Fig. 3 Linear Kernel rappresentation

3.3.2 ***Polynomial Kernel..*** It is a function used to handle data that is not linearly separable. By introducing non-linearity into the data, they are projected into a higher-dimensional feature space where they could potentially change and become linearly separable.

Mathematically, it is expressed as: $K(x, y) = (\gamma \cdot (x \cdot y) + r_0)^d$ where γ is a hyperparameter that controls the decision boundary shape, r is a coefficient controlling the independent term, and d is the degree of the polynomial that influences the level of non-linearity.

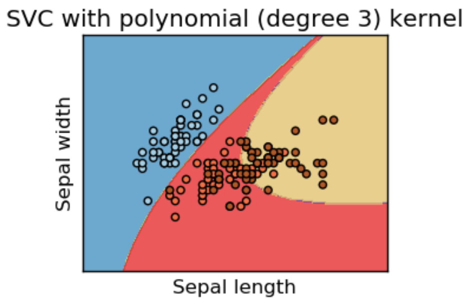

Fig. 4 Polynomial Kernel representation

3.3.3 ***RBF Radial Basis Function..*** It is the most commonly used kernel as it can handle highly complex and non-linear data. This approach introduces a measure of similarity between data points based on their Euclidean distance in the input space. Among the samples, those that are closer to each other will have a higher kernel value, while those farther apart will have a lower kernel value. Mathematically, this is described as: $K(x, y) = \exp(-\gamma \cdot ||x - y||^2)$ where γ is a parameter controlling the data sensitivity to details, and $||x - y||$ is the Euclidean distance between vectors x and y.

3.4 KNN: K-Nearest Neighbors. To solve the classification problem, the **KNN (K-Nearest Neighbors)** algorithm is used. This non-parametric classifier, based on supervised learning, leverages proximity to classify or predict the grouping of an individual point. By arbitrarily choosing a coefficient K, the number of points to consider when measuring Euclidean distances is determined. The algorithm examines the K closest points to the data point being classified and assigns it to a certain class based on the most common class among these K analyzed points. This classification is based on the assumption that similar points can be found close to

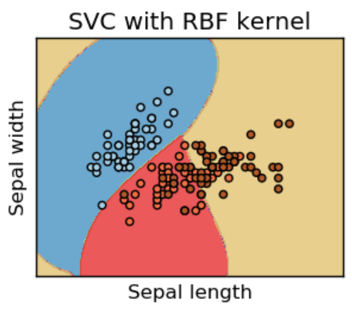

Fig. 5 RBF: Radial Basis Function representation

each other. The selection of K is crucial as it can significantly influence the results, and in case this is too small, it risks losing the ability to generalize.

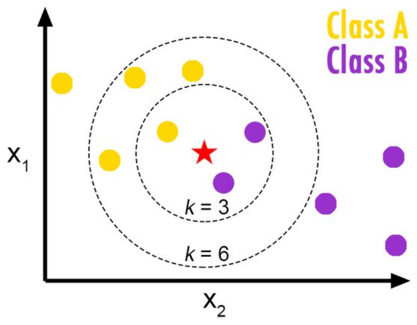

Fig. 6 KNN example

3.5 CNN: Convolutional Neural Networks. This is an effective neural network architecture for working with two-dimensional data, including images, audio signals, and spoken text [8]. In the context of image recognition, CNNs are particularly useful for detection, segmentation, and classification tasks.

A CNN consists of an input layer, an output layer, and many hidden layers in between.

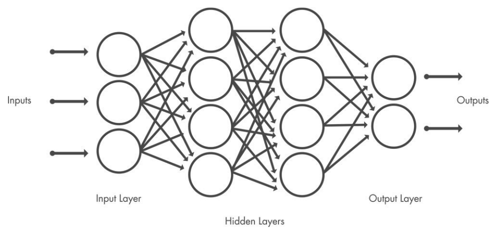

Fig. 7 CNN architecture

These layers internally perform data extraction operations to learn the composition of features. The main layers used are:

- **Convolution Layer**: It performs convolution operations on the input images. These operations are carried out to extract patterns, textures, edges, and features that are relevant from the images.
- **ReLU (Rectified Linear Unit)**: It enables faster and more effective training by mapping negative values to zero and preserving positive ones.
- **Pooling**: It simplifies the output by performing non-linear downsampling, thereby reducing the number of parameters that the network needs to learn. This increases computational efficiency and reduces overfitting.
- **Fully Connected Layer**: This layer attempts to produce class scores from the activations, which are used for classification [8].

After learning features in multiple layers, a CNN architecture transitions to the classification phase. The final layer of the CNN architecture employs a classification layer to provide the output for the final classification.

In this project, a feed-forward neural network with the following characteristics is implemented:

- **3 convolutional layers**: Extracts features from the input.
- **1 max-pooling layer**: Reduces feature dimensionality.
- **2 fully connected layers**: Performs the final classification.
- **2 dropout layers**: Reduces the risk of overfitting during training.

Since the CNN was trained on thousands of images, GPU hardware acceleration (3070 Ti) has been utilized to expedite the training process using the PyTorch library.

3.6 Libraries & technical tools. The development of these methodologies was carried out using **Python**, a high-level object-oriented programming language. This language is very powerful as it is extensively supported by libraries, including those that enable the development of machine learning algorithms. Among the various libraries used, notable ones include:

- **Numpy** is an open-source library that provides support for large matrices and multidimensional arrays, along with a vast collection of high-level mathematical functions.
- **Scikit-learn** is an open-source machine learning library. This tool readily provides programmers with a range of algorithms (including PCA, SVM, KNN), mathematical evaluation methods, data clustering, and feature selection.
- **PyTorch** is an open-source machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing.
- **Matplotlib**s a library for creating charts. It is used to create a wide range of static visualizations, such as line charts, scatter plots, bar charts, and histograms.

4 Experiments & Results

For the experimental phase, the dataset of 3,762 grayscale MRI images was divided into two parts: the **Train set**, consisting of 75% of the images, and the **Test set**, consisting of the remaining 25%. From the **Train set**, another 10% was extracted to create the **Validation set**.

4.1 Evaluation metrics. In order to evaluate the performance of the considered classification models **(KNN - SVM - CNN)**, a **confusion matrix** is calculated for each of them, and a **classification report** is generated.

4.1.1 Confusion matrix. highlights where the model makes errors, in which instances it performs worse, and in which instances it performs better. The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier.

4.1.2 Classification report. is a tool from **scikit-learn** that calculates key performance evaluation metrics for a classification model.

This tool provides:

- **Precision**: Measures how accurately the model classifies examples as positive. It is described as the ratio of true positives to the sum of true positives and false positives.

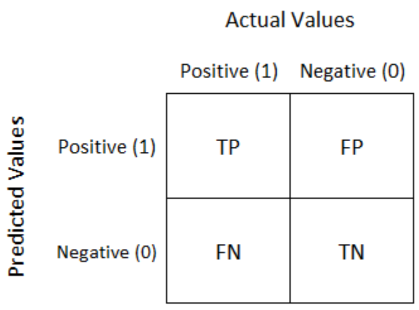

Fig. 8 Two class confusion matrix

- **Recall**: Measures how well the model can capture all true positives. It is defined as the ratio of true positives to the sum of true positives and false negatives.
- **F1-Score**: A harmonic mean of precision and recall.
- **Support**: Represents the number of samples in each class in the test set.
- **Accuracy**: Provides the overall percentage of correct classifications.

All experiments were conducted on a machine with an i7-12700K processor, 32GB of RAM, and an RTX 3070-ti GPU.

4.2 KNN. The training set is used to train the model. The model is tested on the validation set with different **K (1 - 3 - 5 - 7 - 9)**. The **K** that achieves the highest accuracy on the validation set is ultimately used as the parameter to create the model that will be tested on the test set.

K	1	3	5	7	9
Accuracy	95.41%	**95.76%**	92.23%	91.17%	91.52%

Table 1 KNN with different K on Validation Set

The **K** that showed the highest accuracy on the validation set is **K = 3**. The model with **K = 3** on the test set achieved an accuracy of **94%**. Please refer to **Table 2 and 3** for the confusion matrix and the associated classification report.

	Positive (1)	Negative (0)
Tumor	495	31
No tumor	25	390

Table 2 Confusion matrix

	Precision	Recall	F1-score	Support
Non Tumor - 0	0.95	0.94	0.95	526
Tumor - 1	0.93	0.94	0.93	415
Accuracy			0.94	941

Table 3 Classification report of a KNN with K = 3

4.3 PCA + KNN. Using the PCA algorithm with a number of components equal to 2, a drastic drop in classification performance was observed at the expense of execution time, which is approximately **2.06 seconds**, representing a reduction of more than **50%** compared to the KNN algorithm.

The **K = 9**, having the highest accuracy, was used to create the model to be tested on the test set.

From this table, we can observe that the overall performance significantly decreases compared to Table 5. In particular, there is an accuracy of 79%, which is 15% lower than KNN with input data not dimensionally reduced by PCA.

K	1	3	5	7	9
Accuracy	77.39%	80.92%	81.63%	81.63%	**82.33%**

Table 4 PCA + KNN with different K on Validation Set

	Positive (1)	Negative (0)
Tumor	495	31
No tumor	25	390

Table 5 Confusion matrix

	Precision	Recall	F1-score	Support
Non Tumor - 0	0.82	0.83	0.82	526
Tumor - 1	0.78	0.76	0.77	415
Accuracy			0.80	941

Table 6 Classification report on PCA + KNN with K = 9

4.4 SVM. The Support Vector Classifier (SVC) is the classifier that implements the SVM algorithm through **scikit-learn**. Experiments were conducted using three kernels: **Linear, Polynomial, and RBF**.

4.4.1 Linear Kernel. For the experiment with the linear kernel, the model underwent a number of optimizations equal to 100. The model achieves an accuracy of **75%**. In **Table 7 and 8**, the confusion matrix and associated classification report are displayed.

	Positive (1)	Negative (0)
Tumor	382	144
No tumor	89	326

Table 7 Confusion matrix SVC with Linear Kernel

	Precision	Recall	F1-score	Support
Non Tumor - 0	0.81	0.73	0.77	526
Tumor - 1	0.69	0.79	0.74	415
Accuracy			0.75	941

Table 8 Classification report on SVC with Linear Kernel

4.4.2 Polynomial Kernel. The Polynomial kernel achieved an accuracy of **48.99%** on the test set. In **Table 9 and 10**, the confusion matrix and associated classification report are displayed.

	Positive (1)	Negative (0)
Tumor	47	479
No tumor	1	414

Table 9 Confusion matrix SVC with Polynomial Kernel

	Precision	Recall	F1-score	Support
Non Tumor - 0	0.98	0.09	0.16	526
Tumor - 1	0.46	1.00	0.63	415
Accuracy			0.49	941

Table 10 Classification report on SVC with Polynomial Kernel

4.4.3 RBF Kernel. The RBF kernel achieved an accuracy of **70.03%** on the test set. In **Table 11 and 12**, the confusion matrix and associated classification report are displayed.

Among the various kernels, the one that achieved the highest accuracy is the linear kernel with 100 maximum iterations. Due to

	Positive (1)	Negative (0)
Tumor	317	209
No tumor	73	342

Table 11 Confusion matrix SVC with RBF Kernel

	Precision	Recall	F1-score	Support
Non Tumor - 0	0.81	0.60	0.69	526
Tumor - 1	0.62	0.82	0.71	415
Accuracy			0.70	941

Table 12 Classification report on SVC with RBF Kernel

the high dimensionality of the features, the model lost the meaning of relative distances, causing the SVM to struggle in defining decision boundaries and, as a result, not achieving high precision. The high dimensionality of the features is a problem for this type of algorithms as it significantly reduces execution performance.

4.5 PCA + SVM. Applying a dimensionality reduction algorithm like PCA to these types of datasets allows us to use the SVM-based classifier more efficiently. Using 3 principal components as a parameter for PCA, tests were re-run with the three kernels considered **(Linear, Polynomial, RBF)**.

4.5.1 PCA + Linear Kernel. The use of PCA + Linear Kernel led to a drop in performance, decreasing from **75%** to **51.11%**, a loss of **24%**. From the confusion matrix, it can be observed that the model has a high rate of False Positives and False Negatives. In **Table 14 and 15**, the confusion matrix and associated classification report are displayed.

	Positive (1)	Negative (0)
Tumor	287	239
No tumor	221	194

Table 13 Confusion matrix PCA + SVC with Linear Kernel

	Precision	Recall	F1-score	Support
Non Tumor - 0	0.56	0.55	0.56	526
Tumor - 1	0.45	0.47	0.46	415
Accuracy			0.51	941

Table 14 Classification report on PCA+SVC with Linear Kernel

4.5.2 PCA + Polynomial Kernel. The use of PCA + Polynomial Kernel led to a drastic increase in performance, going from an accuracy rate of **48.99%** to **80.87%** with an increase of **31.88%**. In **Table 16 and 17**, the confusion matrix and associated classification report are displayed.

	Positive (1)	Negative (0)
Tumor	381	145
No tumor	35	380

Table 15 Confusion matrix PCA + SVC with Linear Kernel

	Precision	Recall	F1-score	Support
Non Tumor - 0	0.92	0.72	0.81	526
Tumor - 1	0.72	0.92	0.81	415
Accuracy			0.81	941

Table 16 Classification report on PCA+SVC with Polynomial Kernel

4.5.3 PCA + RBF Kernel. The use of PCA + RBF Kernel led to a increase in performance, going from an accuracy rate of **70.03%** to **87.35%** with an increase of **17.32%**. From the confusion matrix, it can be observed that the model has balanced False Positive and False Negative rates. In **Table 18 and 19**, the confusion matrix and associated classification report are displayed.

	Positive (1)	Negative (0)
Tumor	465	61
No tumor	58	357

Table 17 Confusion matrix PCA + SVC with Linear Kernel

	Precision	Recall	F1-score	Support
Non Tumor - 0	0.89	0.88	0.89	526
Tumor - 1	0.85	0.86	0.86	415
Accuracy			0.87	941

Table 18 Classification report on PCA + SVC with RBF Kernel

After utilizing PCA to reduce dimensionality, the experiments demonstrate that the **RBF Kernel** is the most accurate on the test set (**87.35%**). It is also evident that dimensionality reduction through PCA played a crucial role in terms of classification accuracy and processing time. The execution time has significantly decreased, as can be seen in **Table 20** (the lower the execution time the better).

Kernel	Seconds
Linear Kernel	48.48
PCA + Linear Kernel	**0.071**
Polynomial Kernel	56.80
PCA + Polynomial Kernel	**0.073**
RBF Kernel	62.34
PCA + RBF Kernel	**0.07**

Table 19 Comparison of time performance on Kernels with and without PCA

5 CNN Convolutional Neural Network

In this experiment, the CNN was trained for 100 epochs, and on the test set, it achieved an accuracy of **95.34%**.

	Positive (1)	Negative (0)
Tumor	508	18
No tumor	26	389

Table 20 Confusion matrix of the CNN

The model has a recall of **93.7%** and a precision of **95.5%**. In **Table 21 and 22**, you can find the confusion matrix and the trend of loss / accuracy during the epochs.

Epoch	Loss	Accuracy
0	0.18476	0.58
20	0.14416	0.875
40	0.01653	0.935
60	0.08495	0.964
80	0.02391	0.993

Table 21 Loss Trend Over Epochs

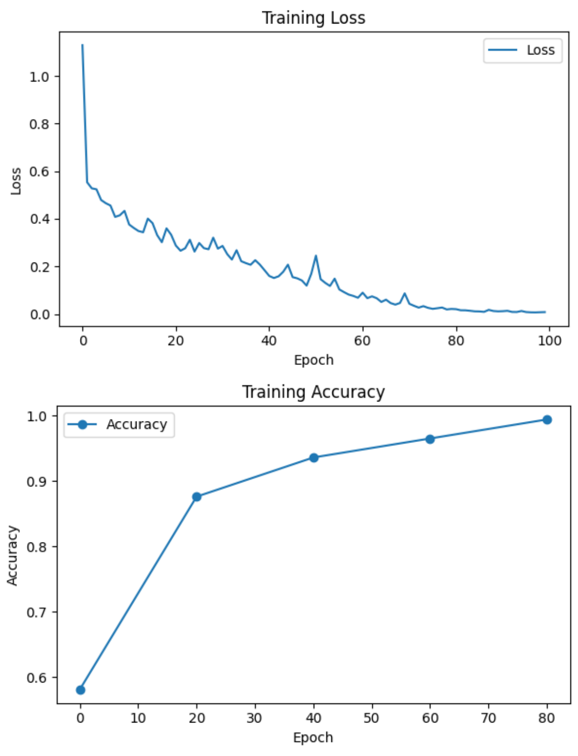

Fig. 9 Plots of training accuracy and loss trends over epochs

6 Conclusions

In this paper, the performance of three algorithms (KNN, SVM, CNN) has been analyzed under different conditions. The results show that the optimal approach in terms of accuracy is achieved through a CNN (Convolutional Neural Network), with an accuracy of **95.34%**.

The KNN approach, with a **K = 3**, remains an excellent binary classifier for this dataset, achieving an accuracy of **94%**. The SVM-based classifier has shown uncertainty due to the high dimensionality of the data, achieving an accuracy of **75%** with a Linear Kernel. This uncertainty was partially addressed by combining the SVM classifier with dimensionality reduction through PCA, resulting in an increased accuracy of **87.35%** with an RBF Kernel.

This paper demonstrates the effective utility of these methods as tools to assist medical professionals in diagnosing the presence or absence of brain tumors. As future work, it will undoubtedly be necessary to make the classifier capable not only of detecting the presence (or absence) of a tumor, but also of determining its type (e.g. Gliomas, Meningiomas, Glioblastomas, Cysts, Acoustic Neuromas).

References

[1] Martucci, M., Russo, R., Schimperna, F., D'Apolito, G., Panfili, M., Grimaldi, A., Perna, A., Ferranti, A. M., Varcasia, G., Giordano, C., and Gaudino, S., 2023, "Magnetic Resonance Imaging of Primary Adult Brain Tumors: State of the Art and Future Perspectives," Biomedicines, **11**(2).

[2] Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., and Paluri, M., 2018, "A Closer Look at Spatiotemporal Convolutions for Action Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[3] Chatterjee, S., Nürnberger, A., and Speck, O., 2022, "Classification of brain tumours in MR images using deep spatiospatial models," Scientific Reports, **12**.

[4] Jolliffe IT, C. J., 2016, "Principal component analysis: a review and recent developments," Philos Trans A Math Phys Eng Sci.

[5] Hsu, C.-W. and Lin, C.-J., 2002, "A comparison of methods for multiclass support vector machines," IEEE Transactions on Neural Networks, **13**(2), pp. 415–425.

[6] Cortes, C. and Vapnik, V. N., 1995, "Support-Vector Networks," Machine Learning, **20**, pp. 273–297.

[7] Patle, A. and Chouhan, D. S., 2013, "SVM kernel functions for classification," *2013 International Conference on Advances in Technology and Engineering (ICATE)*, pp. 1–9, doi: 10.1109/ICAdTE.2013.6524743.

[8] O'Shea, K. and Nash, R., 2015, "An Introduction to Convolutional Neural Networks," CoRR, **abs/1511.08458**.

List of Figures

List of Tables